Seshat: A tool for managing and verifying annotation campaigns of audio data

Hadrien Titeux, Rachid Riad, Xuan-Nga Cao, Nicolas Hamilakis, Kris Madden, Alejandrina Cristia, Anne-Catherine Bachoud-Lévi, Emmanuel Dupoux

Seshat: A tool for managing and verifying annotation campaigns of audio data

Hadrien Titeux[⋆1] **, Rachid Riad**[⋆1,2]**, Xuan-Nga Cao**[1]**, Nicolas Hamilakis**[1]**, Kris Madden**[1]**, Alejandrina Cristia**[1]**, Anne-Catherine Bachoud-Lévi**[2]**, Emmanuel Dupoux**[1]

1 LSCP/ENS/CNRS/EHESS/INRIA/PSL Research University, Paris, France

2 NPI/ENS/INSERM/UPEC/PSL Research University, Créteil, France

{hadrien.titeux, rachid.riad}@ens.fr,

{ngafrance, nick.hamilakis562, thekrismadden, alecristia, bachoud, emmanuel.dupoux}@gmail.com

Abstract

We introduce Seshat, a new, simple and open-source software to efficiently manage annotations of speech corpora. The Seshat software allows users to easily customise and manage annotations of large audio corpora while ensuring compliance with the formatting and naming conventions of the annotated output files. In addition, it includes procedures for checking the content of annotations following specific rules that can be implemented in personalised parsers. Finally, we propose a double-annotation mode, for which Seshat computes automatically an associated inter-annotator agreement with the γ measure taking into account the categorisation and segmentation discrepancies.

Keywords: speech transcription, speech corpora, annotations management

1. Introduction

Large corpora of speech, obtained in the laboratory and in naturalistic conditions, become easier to collect. This new trend broadens the scope of scientific questions on speech and language that can be answered. However, this poses an important challenge for the construction of reliable and usable annotations. Managing annotators and ensuring the quality of their annotations are highly demanding tasks for research endeavours and industrial projects (Zue et al., 1990). When organised manually, the manager of annotation campaigns usually faces three major problems: the *mishandling of files* (e.g., character-encoding problems, incorrect naming of files), the *non-conformity of the annotations* (Moreno et al., 2000), and the *inconsistency of the annotations* (Gut and Bayerl, 2004).

In this paper, we introduce *Seshat*, a system for the automated management of annotation campaigns for audio/speech data which addresses these challenges. It is built on two components that communicate via a Restful API: a back-end (server) written in Flask and a front-end (client) in Angular Typescript. Seshat is easy to install for non-developers and easy to use for researchers and annotators while having some extension capabilities for developers.

In Section 2, we describe the related work on annotations tools, which do not provide solutions to all the aforementioned challenges during corpus creation. In Section 3, we make an overview of the different functionalities of the software. Then, we explain, in Section 4, the architecture of the software, and also the several UX/UI design and engineering choices that have been made to facilitate the usage of the platform. We describe how to use of Seshat in Section 5 and Section 6 presents two specific use-cases. Finally, we conclude and describe future plans for Seshat in Section 7.

⋆ Equal contribution. This work was conducted while E. Dupoux was a part-time Research Scientist at Facebook AI Research. Code for Seshat is available on Github at `https://github.com/bootphon/seshat`

2. Related Work

Self-hosted annotation systems. There are many standalone solutions for the transcription of speech data that are already used by researchers: Transcriber (Barras et al., 2001), Wavesurfer (Sjölander and Beskow, 2000), Praat (Boersma and others, 2002), ELAN (MacWhinney, 2014), XTrans (Glenn et al., 2009). These systems allow the playback of sound data and the construction of different layers of annotations with various specifications, with some advanced capabilities (such as annotations with hierarchical or no relationship between layers, number of audio channels, video support). Yet, these solutions lack a management system: each researcher must track the files assigned to annotators and build a pipeline to parse (and eventually check) the output annotation files. Moreover, checking can only be done once the annotations have been submitted to the researchers. This task becomes quickly untraceable as the number of files and annotators grow. In addition, most of these transcription systems do not provide a way to evaluate consistency (intra- and inter-annotator agreement) that would be appropriate for speech data (Mathet et al., 2015).

Web-based annotations systems. There are several web-based annotation systems for the annotation of audio data. Among them we find light-weight systems, like the VIA software (Dutta and Zisserman, 2019) or Praat on the web (Dominguez et al., 2016) that allow to build simple layers of annotations. However, they do not provide a proper management system for a pool of annotators nor do they integrate annotation checking.

On the other side of the spectrum, there are more sophisticated systems with various capabilities. Camomille (Poignant et al., 2016) and the EMU-SDMS system (that can also be used offline) (Winkelmann et al., 2017) allow to work with speech data and to distribute the tasks to several annotators. But these systems require expertise in web hosting and technologies to deploy and modify them.

Finally, WebAnno (Yimam et al., 2013) and GATE Teamware (Bontcheva et al., 2013) are the tools that most closely match our main contributions regarding quality con-

trol (conformity and consistency checking), annotators' management and flexibility. WebAnno includes consistency checking with the integration of different metrics (Meyer et al., 2014). However, these tools have only been built for text data. The format and all the custom layers have been designed for Natural Language Processing tasks. Porting WebAnno to support speech data seemed a major engineering challenge. That is why it appeared necessary to develop a new and user-friendly tool addressed to the speech community.

3. Overview of Seshat

Seshat is a user-friendly web-based interface whose objective is to smoothly manage large campaigns of audio data annotation, see Figure 2. Below, we describe the several terms used in Seshat's workflow:

Audio Corpus
: A set of audio/speech files that a **Campaign Manager** wants to annotate. It is indicated either by a folder containing sound files, or by a CSV summarizing a set of files. We support the same formats as Praat so far: WAV, Flac and MP3.

Annotation Campaign
: An object that enables the **Campaign Manager** to assign **Annotation Tasks** to the **Annotators**. It references a **Corpus**, and allows the Manager to track the annotation's tasks progress and completion in real time. At its creation, a **Textgrid Checking Scheme** can also be defined for that campaign.

Annotation Task
: It is contained in an **Annotation Campaign**, it references an audio file from the campaign's designated **Audio Corpus**, and assigned to **Annotators**. It can either be a *Single Annotator Task* (assigned to one Annotator) or a Double Annotator Task (assigned to two annotators, who will annotatote the assigned task in parallel).

Textgrid Checking Scheme
: A set of rules defining the TextGrid files' structure and content of the annotations. It is set at the beginning of the **Annotation Campaign's** creation, and is used to enforce that all TextGrids from the campaign contain the same amount of Tiers, with the same names. It can also enforce, for certain chosen tiers, a set of valid annotations.

Campaign Manager
: Users with the rights to create **Annotation Campaigns** and **Annotators** user accounts, and assign **Annotation Tasks** to **Annotators**.

Annotator
: Users who are assigned a set of **Annotation Tasks**. Their job is to complete the annotation of the audio files with the Praat software.

  If the TextGrid file they submit does not comply with their **Annotation Task**'s **TextGrid Checking Scheme**, Seshat pinpoint their annotation errors with detailed messages. The annotator can resubmit the concerned file to the platform based on these different feedbacks.

Once they they connected to their instance of Seshat, *campaign managers* can access ongoing annotation campaigns or create new ones. Campaign managers are able to add *annotators*, assign *annotation tasks* and track progress. Annotator see a list of assigned tasks. The first step for them is to download the sound file with its corresponding autogenerated template TextGrid. In the current implementation, the annotation work has to be done locally with Praat. An upcoming version will use of web tools like Praat on the web (Dominguez et al., 2016). Once the task is completed, the TextGrid file is to be uploaded to Seshat via the web interface. We used the TextGrid format because of the wide acceptance of the Praat software in the speech science community (e.g., language acquisition research, clinical linguistics, phonetics and phonology).

The *Textgrid Checking Scheme* that encompasses rules on the tier's naming, file structure, and the content of the annotations, is associated with a specific campaign and defined at the creation of the campaign. Seshat back-end will automatically check that the submitted TextGrid file conforms to the *Annotation Campaign*'s *Textgrid Checking Scheme*.

Seshat allows the campaign manager to create two type of tasks: single annotator, and double annotator. Regarding the first task, one audio file is attributed to one annotator. Once the annotation is completed, Sesha automatically checks the conformity of the annotation, and only declares a tasks completed if the conformity checks is passed. Regarding the second task, one audio file is attributed to two annotators. The two annotators annotate the same file *independently*, then the two versions are merged and the annotators are guided through a *compare and review* process to agree one final version. We summarise in the Figure 1 the different steps for the double-annotator task. At each step during merging, the two annotators are provided feedbacks to focus on where are the disagreements. This process also results in the computation of an *Inter-annotator agreement* for each file. The double annotator task can be used to train new annotators alongside experts.

Annotating speech data is a joint task of segmentation and categorisation of audio events. That is why we adopted the γ measure (Mathet et al., 2015) to evaluate the inter- or intra- annotator agreement in each individual tier. Campaign manager can customise the distance used by γ by inserting a custom distance along their own parser (See short snippet of code for a parser of French Phonetics with the SAMPA alphabet in Algorithm 1).

4. Development

4.1. Engineering choices

Our utmost priority when building Seshat was to make it as easy as possible for others to deploy, use, administer and eventually contribute to. To do so, we chose the most common frameworks that are free and open-source, all of which are detailed in the following sections. Additionally, to match the current trend in web development, we decided to use the so-called "web-app" architecture for Seshat, i.e.,

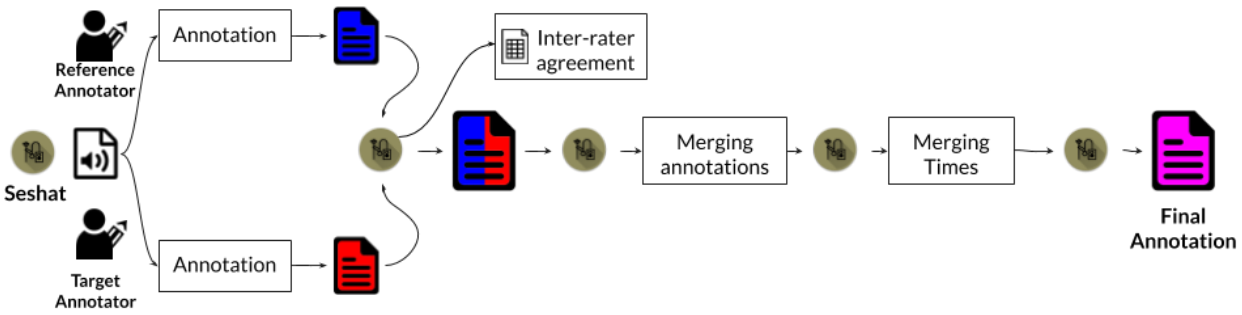

Figure 1: Double Annotator task overview. Inter-rater agreement is computed by the interface for the first independently annotated files in Red and Blue.

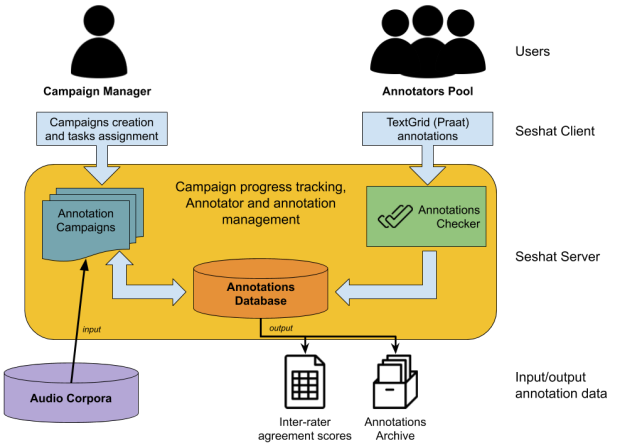

Figure 2: Seshat architecture: two different front-end, for annotators and campaign manager, a back-end with persistent data storage of the annotations and the inter-rater agreements.

we separated the application into two distinct entities: a *front-end*, running on the browser, and a *back-end*, serving data to the front-end and interacting with the database.

4.1.1. Back-end Choices

The back-end system runs on a server. It holds and updates the campaign databases and runs the annotation checking and inter-rater agreement evaluation services. We chose Python, given its widespread use in the scientific community[1], with a wide array of speech and linguistic packages. Moreover, its usage on the back-end side will allow the future integration of powerful speech processing tools like Pyannote (Bredin et al., 2019) to semi-automatize annotations. We thus went for Python3.6 for Seshat's server back-end. We used the Flask-Smorest[2] extension (which is based on Flask[3]) to clearly and thoroughly document our API, which can be exported to the popular OpenAPI 3.0.2[4] RESTful API description format.

The files and server data are stored on a MongoDB[5] database, chosen for its flexible document model and general ease of use. We used the Object-Relational Mapping (ORM) MongoEngine[6] to define our database schemas and interact with that database. MongoDB's GridFS system also allowed us to directly store annotation files (which are usually very light-weight) directly in the database, instead of going through the file system.

4.1.2. Front-end Choices

The front-end handles all of the interactions between the users (campaing manager or annotator) with the databses. It is implemented as an App within their browser. We decided to base Seshat's front-end on the Angular Typescript [7] framework. Despite its' steep learning curve, it enforces strict design patterns that guarantee that others can make additions to our code without jeopardising the stability of the App. Angular Typescript has a wide community support in the web development industry and is backed by Google and Microsoft. Moreover, the fact that it is based on TypeScript alleviates the numerous shortcomings of JavaScript, ensuring our implementation's readability and stability.

4.2. UX/UI Choices

The interface and the features we selected for our implementation are the process of a year-long iterative process involving a team of annotators, two campaign managers and software engineers. We followed some guiding principles from the recent Material[8] design language. Our goal while designing our interface (with the help of a professional designer) was to make it fully usable by non-technical people. We also put some extra care into the annotators' interface to give them a clear sense of what is to be done, how they should follow the annotation protocol, and how to correct potential errors in their annotations (See Figure 3) The goal was to reduce the number of actions to perform for annotators and enable to focus only on the annotations content.

[1] https://insights.stackoverflow.com/survey/2019

[2] https://github.com/marshmallow-code/flask-smorest

[3] https://www.palletsprojects.com/p/flask/

[4] https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.2.md

[5] https://www.mongodb.com/

[6] http://mongoengine.org/

[7] https://angular.io/

[8] https://material.io/design/introduction/#principles

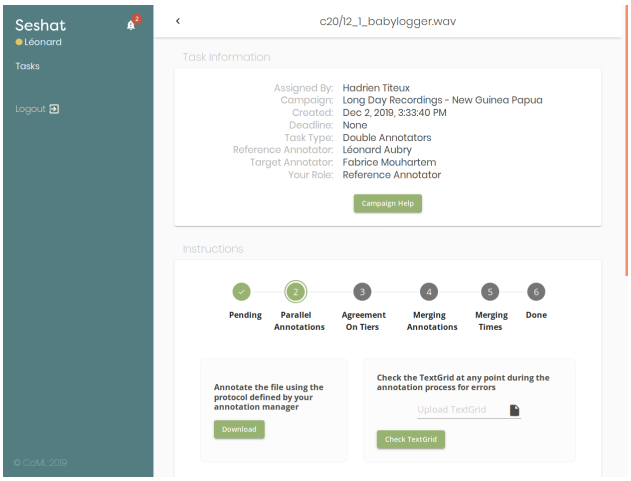

Figure 3: Assigned task from the annotator's point of view.

5. Using Seshat

5.1. Installation and Setup

Setting up a modern fully-fledged web service is a arduous task, usually requiring a seasoned system administrator as well as sometimes having very precise system requirements. Luckily, the Docker[9] virtualisation platform ensures that anyone with a recent-enough install of that software can set up Seshat in about one command (while still allowing some flexibility via a configuration file). For those willing to have a more tightly-controlled installation of Seshat on their system, we also fully specify the manual installation steps in our online documentation[10]).

Importing an audio corpus that you are willing to annotate is easy as dropping files into a default 'corpora/' folder. It is possible to either drop a folder containing audio files (with no constraints on the folder's structure), or a CSV file listing audio filenames along with their durations (in case the files are sensitive and you're not willing to risk them being hosted on the server). It is then possible to review the automatically imported files *via* the web interface.

5.2. Launching and monitoring an annotation campaign

The Campaign manager can easily define and monitor annotation campaign. As shown in Figure 5, the online form enable to choose corpora, pre-define and pre-configure the annotations scheme (tiers and parsers). There are 2 types of tiers already implemented by default: one with no check at all, and one with pre-defined categories. For the latter, these categories are pre-defined when the campaign is created.

Only Campaign managers can access and build new campaigns. If Campaign manager have several campaigns they can easily switch between them via the menu bar or get a full overview with the dashboard (See Figure 4). The campaign managers can visualise the progress of the assigned tasks at the campaign level or more precisely at the task level. They can retrieve all the intermediate files that have been created for each task. For instance, the campaign manager can examine qualitatively and quantitatively what are the annotation differences before the merge phases of the double annotator task.

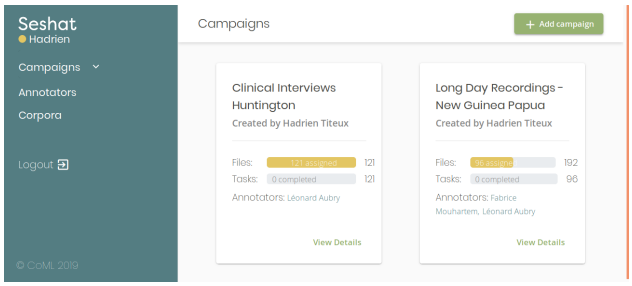

Figure 4: Dashboard for the campaign manager

5.3. Scripting API

For those willing to interact with Seshat using code, it is possible to interact with Seshat using either its RESTful API or its command-line interface (CLI). The API endpoints that can be called are all listed in a simple interface, and can be made from any programming language able to make HTTP requests. The CLI interface can be used via your terminal, and therefore can be interacted with using Bash scripts.

A typical usage of these features would be to assign annotation tasks from a large speech corpus (spoken by several speakers) to a large pool of annotators, all the while making sure each annotator has a similar number of tasks, with each speaker being evenly distributed among annotators as well. This would be tedious to do manually via the user interface, but easy to program in any scripting language.

5.4. Annotation Parser Customisation

We aimed at a reasonable trade-off between simplicity and flexibility for the TextGrid annotations checking component. However, we understand (from our own experience in particular) that sometimes annotations can follow a very specific and complex standard (for instance, parsing SAMPA phonemes strings). To allow users to define their own annotation standards, we added the possibility for users to define an annotation parser, via a simple package-based extension system (taking inspiration from pyannote's extension system). Anyone willing to create a new annotation parser has to be able to program in Python and have a minimal understanding of its packaging system.

As presented in our example French SAMPA Parser (Algorithm 1), implementing a custom annotation parsers only requires the overload of two methods from Seshat's `BaseCustomParser` class:

- `check-annotation`: takes an annotation string as input and raises an error if and only if the annotation is deemed to be invalid. It doesn't return anything.
- `distance`: takes two annotations as input and should return a float corresponding to the distance between these two annotations.

[9] https://www.docker.com/

[10] https://seshat-annotation.readthedocs.io/

```python
class FrenchSAMPAParser(BaseCustomParser):
    PHONEMES = ['&/', '2','9','9~','@','A','A/','E','E/','H',
                'N','O','O/','R','S','U~/','Z','a','a~','b',
                'd','e','e~','f','g','i','j','k','l','m','n',
                'o','o~','p','s','t','u','v','w','y','z','J']

    def __init__(self):
        # sorting (descending) phonemes by length
        self.PHONEMES = sorted(self.PHONEMES, key=len, reverse=True)

    def parse_sampa(self, pho_str: str) -> List[str]:
        """Parses a French phoneme string into a phoneme list
         ex : "septa~br" -> [s, e, p, t, a~, b, r]"""
        original_str = str(pho_str)
        pho_list = []
        while pho_str:
            parsed_phoneme = False
            # trying to match longest phoneme names first
            for phoneme in self.PHONEMES:
                if pho_str[:len(phoneme)] == phoneme:
                    pho_list.append(pho_str[:len(phoneme)])
                    pho_str = pho_str[len(phoneme):]
                    parsed_phoneme = True
                    break
            if not parsed_phoneme:
                raise AnnotationError(
                    "Can't parse phonetic form %s (stuck at %s)"
                                    % (original_str, pho_str))
        return pho_list

    def check_annotation(self, annot: str) -> None:
        self.parse_sampa(annot)

    def distance(self, annot_a: str, annot_b : str) -> float:
        """Computes the levenshtein distance between two phone sequences"""
        parsed_a = self.parse_sampa(annot_a)
        parsed_b = self.parse_sampa(annot_b)
        return levenshtein(parsed_a, parsed_b)
```

Algorithm 1: Parser Plugin Example. This parser checks the units to allow only phone sequences in the SAMPA format. The distance that can be used for the inter-rater agreement is the Levenshtein distance.

5.5. Inter-rater agreement: the γ measure

It is necessary have a measure of confidence to obtain high-quality datasets and therefore to draw valid conclusions from annotations. Annotations tasks of audio and speech data usually have some specificities. The items to annotate have to be both segmented in time and categorised. The segments can be hierarchically defined or overlapping. In addition, the audio stream may require only sparse annotations (especially in-the-wild recordings which contain a lot of non-speech segments). To evaluate speech annotations, the measure needs to take these characteristics into account. That is why we decided to re-implement and compute the γ measure (see Mathet et al. (2015) for its design and the advantages of this measure over previous agreement measures).
First, the γ software aligns (tier-wise) the annotations of the different annotators. To align the two sets of annotations the γ measure the distance between all the individual units. The difference of position of two annotated units u and v is measured with the positional distance:

$$d_{\text{pos}}(u,v) = \left(\frac{|start(u) - \text{start}(v)| + |\text{end}(u) - \text{end}(v)|}{(end(u) - \text{start}(u)) + (end(v) - \text{start}(v))} \right)^2$$

If the tiers are categorical, the distance for the content of the annotated units u and v is defined as:

$$d_{\text{cat}}(u,v) = \mathbb{1}(cat(u) == cat(v))$$

This distance can be over-written by the custom parser as mentioned above. These two distance are summed with equal weights to obtain the distance between every annotated units from 2 annotators. Then, it is possible to obtain the *disorder* $\delta(a)$ of a specific alignment a by summing the distance of all the aligned units in a. All possible alignments a are considered and the one that minimises the disorder $\delta(a)$ is kept.
To get the value of γ, the disorder is chance-corrected to obtain an expected disorder. It is obtained by re-sampling randomly the annotations of the annotators. This means that real annotations are drawn from the annotators, and one position in the audio is randomly chosen. The annotation is split at this random position and the two parts are permuted. It is then possible to obtain an approximation of the expected disorder δ_e. The final agreement measure is defined as:

$$\gamma = 1 - \frac{\delta(a)}{\delta_e}$$

This γ measure is automatically computed by the back-end server for the double-annotator tasks. The Campaign manager can retrieve these measures in Seshat by downloading a simple CSV file.

6. Use cases

We present two use cases on which Seshat was developped: clinical interviews, and daylong child-centered recordings.

6.1. Clinical interviews

Seshat was intially developped to study the impact of Huntington's Disease (Walker, 2007) on speech and language production. One hundred and fifty two interviews between a neuropsychologist and a patient with the Huntington's Disease (HD) were recorded between June 2018 and November 2019. The campaign manager created a campaign with multiple tiers to annotate the turn takings and the speech/non speech boundaries of the utterances of the patient. For both tasks, the annotations did not need to cover completely the audio (sparsity property mentioned above). For the Turn-taking annotations, there are 3 pre-defined tiers, each one with a single class ('Patient', 'Non-Patient', and 'Noise'), which results in possible overlap between these classes. For the Utterance annotations, there is only one pre-defined class ('Utterance').
To this date, a total of 67 files have been fully annotated with the help of Seshat by a cohort of 18 speech pathologist students (see Figure 5). Among these, 16 have been done by 2 different annotators independently with the Double-annotator task. The results are summarised in Table 1.
Even though there are more categories for Turn-Takings than Utterance (Gut and Bayerl (2004) reported that the more categories the more the task is difficult in speech annotations), the mean γ for the Turn-Takings $\gamma = 0.64$ is slightly higher than the one for Utterance $\gamma = 0.61$. And the range of values for the Turn-Takings is smaller than the Utterance. Indeed, the speech pathologists reported the difficulty to annotate the boundary of utterances in spontaneous speech, with several ambiguous cases due to pauses. These results will help us to redefine the protocol and be more precise on the given instructions.

6.2. In-the-wild child-centered recordings

The Seshat software is also currently used to annotate audio files in a study of day-long audio-recordings captured by

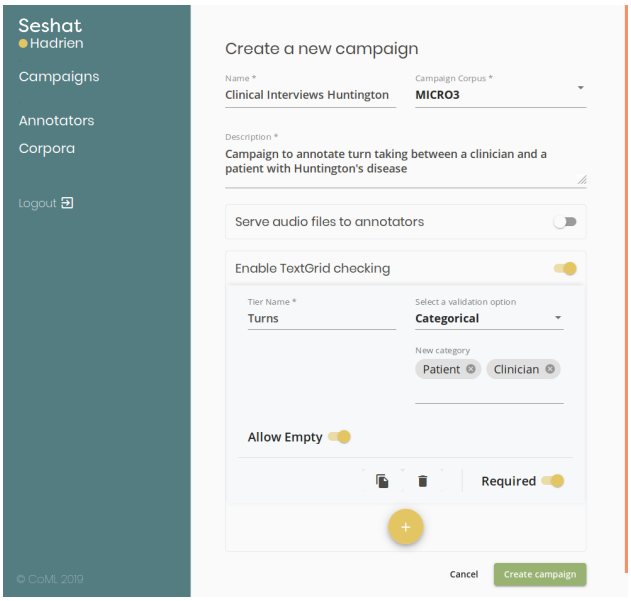

Figure 5: Annotation Campaign definition in Seshat for clinical interviews between a patient with the Huntington's Disease and a neuropsychologist

Tiers	γ		
	Mean	Range	#classes
Turn-Takings	0.64	0.18	3
Utterance	0.61	0.39	1

Table 1: γ Inter-rater agreements summary for 16 clinical interviews between a neuropsychologist and a patient with the HD.

two devices (LENA (Gilkerson and Richards, 2008), and a BabyCloud baby-logger device) worn by young children growing up in remote Papua New Guinea. The project aims at establishing language input and outcomes in this seldom-studied population. To establish reliability levels, 20 1-min files were double-annotated by 2 speech pathology students. Among the tasks given to the annotators there was: (1) locating the portions of Speech (Speech activity), (2) locating the speech produced by an adult that is directed to a child or not (*Adult-Directed Speech* versus *Child-Directed Speech*). As in the previous example, the annotations do not need to cover the full audio file. The Speech Activity task has only 1 class ('Speech') and the Addressee task has 2 classes ('ADS', 'CDS').

Tiers	γ		
	Mean	Range	#classes
Speech activity	0.46	0.60	1
ADS vs CDS	0.27	0.39	2

Table 2: γ Inter-rater agreement for 20 1-min slices extracted from child-centered day-long recordings. ADS and CDS stand for Adult-Directed Speech and Child-Directed Speech respectively.

These recordings have been done in naturalistic and noisy conditions; moreover, the annotators do not understand the language. Probably as a result of these challenges, agreement between annotators is lower than in the Clinical interviews use case. This information is nonetheless valuable to the researchers, as it can help them appropriately lower their confidence in the ensuing speech quantity estimates.

7. Conclusion and Future work

Seshat is a new tool for the management of audio annotation efforts. Seshat enables users to define their own campaign of annotations. Based on this configuration, Seshat automatically enforces the format of the annotations returned by the annotators. Besides, we also add the capability to finely tailor the parsing of the annotations. Finally, Seshat provides automatic routines to compute the inter-rate agreements that are specifically designed for audio annotations. Seshat lays some foundations for more advanced features, either for the interface or the annotation capabilities. In future work, we plan to implement an automatic task assignments and an integration of a diarization processing step to reduce human effort. Another planned feature is to add possibility for the campaign manager to design more complex annotation workflows such as, for instance, dependencies between tiers or more intermediate steps of annotations.

8. Acknowledgements

This research was conducted thanks to Agence Nationale de la Recherche (ANR-17-CE28-0007 LangAge, ANR-16-DATA-0004 ACLEW, ANR-14-CE30-0003 MechELex, ANR-17-EURE-0017, ANR-10-IDEX-0001-02 PSL*, ANR-19-P3IA-0001,) and grants from Facebook AI Research (Research Grant), Google (Faculty Research Award), and Microsoft Research (Azure Credits and Grant), and a J. S. McDonnell Foundation Understanding Human Cognition Scholar Award.

9. References

Barras, C., Geoffrois, E., Wu, Z., and Liberman, M. (2001). Transcriber: development and use of a tool for assisting speech corpora production. *Speech Communication*, 33(1-2):5–22.

Boersma, P. et al. (2002). Praat, a system for doing phonetics by computer. *Glot international*, 5.

Bontcheva, K., Cunningham, H., Roberts, I., Roberts, A., Tablan, V., Aswani, N., and Gorrell, G. (2013). Gate teamware: a web-based, collaborative text annotation framework. *Language Resources and Evaluation*, 47(4):1007–1029.

Bredin, H., Yin, R., Coria, J. M., Gelly, G., Korshunov, P., Lavechin, M., Fustes, D., Titeux, H., Bouaziz, W., and Gill, M.-P. (2019). pyannote. audio: neural building blocks for speaker diarization. *arXiv preprint arXiv:1911.01255*.

Dominguez, M., Latorre, I., Farrús, M., Codina-Filba, J., and Wanner, L. (2016). Praat on the web: an upgrade of praat for semi-automatic speech annotation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 218–222.

Dutta, A. and Zisserman, A. (2019). The via annotation software for images, audio and video. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 2276–2279. ACM.

Gilkerson, J. and Richards, J. A. (2008). The lena natural language study.

Glenn, M. L., Strassel, S. M., and Lee, H. (2009). Xtrans: A speech annotation and transcription tool. In *Tenth Annual Conference of the International Speech Communication Association*.

Gut, U. and Bayerl, P. S. (2004). Measuring the reliability of manual annotations of speech corpora. In *Speech Prosody 2004, International Conference*.

MacWhinney, B. (2014). *The CHILDES project: Tools for analyzing talk, Volume II: The database*. Psychology Press.

Mathet, Y., Widlöcher, A., and Métivier, J.-P. (2015). The unified and holistic method gamma (γ) for inter-annotator agreement measure and alignment. *Computational Linguistics*, 41(3):437–479.

Meyer, C. M., Mieskes, M., Stab, C., and Gurevych, I. (2014). Dkpro agreement: An open-source java library for measuring inter-rater agreement. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 105–109.

Moreno, A., Lindberg, B., Draxler, C., Richard, G., Choukri, K., Euler, S., and Allen, J. (2000). Speechdat-car. a large speech database for automotive environments. In *LREC*.

Poignant, J., Budnik, M., Bredin, H., Barras, C., Stefas, M., Bruneau, P., Adda, G., Besacier, L., Ekenel, H., Francopoulo, G., et al. (2016). The camomile collaborative annotation platform for multi-modal, multi-lingual and multi-media documents. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1421–1425.

Sjölander, K. and Beskow, J. (2000). Wavesurfer-an open source speech tool. In *Sixth International Conference on Spoken Language Processing*.

Walker, F. O. (2007). Huntington's disease. *The Lancet*, 369(9557):218–228.

Winkelmann, R., Harrington, J., and Jänsch, K. (2017). Emu-sdms: Advanced speech database management and analysis in r. *Computer Speech & Language*, 45:392–410.

Yimam, S. M., Gurevych, I., de Castilho, R. E., and Biemann, C. (2013). Webanno: A flexible, web-based and visually supported system for distributed annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–6.

Zue, V., Seneff, S., and Glass, J. (1990). Speech database development at mit: Timit and beyond. *Speech communication*, 9(4):351–356.